A Power Efficient, Oblivious, Path-Diverse, Minimal Routing for Mesh-based Networks-on-Chip

El Sayed M. Saad[1], Sameh A. Salem[2], Medhat H. Awadalla[3], and Ahmed M. Mostafa[4]

[1] Communication, Electronics and Computers Department, Faculty of Engineering, Helwan University, Helwan, Egypt

[2] Communication, Electronics and Computers Department, Faculty of Engineering, Helwan University, Helwan, Egypt

[3] Communication, Electronics and Computers Department, Faculty of Engineering, Helwan University, Helwan, Egypt

[4] Communication, Electronics and Computers Department, Faculty of Engineering, Helwan University, Helwan, Egypt

Abstract

One of the most trade-off aspects in the design of NoCs is the improvement of the network performance, in terms of throughput and latency, while minimizing power consumption. 2D-mesh has become the preferred topology, since it offers low and constant link delay. This paper proposes a Power efficient, Oblivious, Path-diverse, Minimal routing (POPM) for mesh-based Networks-on-Chip. In order to improve the performance of the network, POPM makes routing decisions locally at each hop rather than establishing a fixed and deterministic path between the source and destination nodes. POPM routes each packet separately through a path selected from among all minimal paths. Detailed simulations on a set of synthetic traffic patterns as well as a real application traffic pattern show that POPM competes favorably to existing routing algorithms, including dimension-ordered (DOR) routing, North-Last turn model, and PROM routing.

Keywords: *Network-on-Chip, Minimal Routing, Oblivious routing, Performance, Power.*

1. Introduction

As technology moves towards multi-core system-on-chips (SoCs), networks-on-chip (NoCs) [1-2] are emerging as the scalable fabric for interconnecting the cores. They consist of routers, links, and well-defined network interfaces. Packet-switched interconnection networks [3] facilitate communication between cores by routing packets between them. The structured and localized wiring of such a NoC design simplifies timing convergence and enables robust design that scales well with device performance. One of the key issues in the design of NoCs is the reduction of both area and power dissipation. Such requirements impose important design choices like the topology, switching technique, routing algorithm and the architectural implementation. As a result, most of current NoCs implement regular network topologies that can be easily laid out on a chip surface. Two-dimensional meshes have become the preferred topologies, since they offer low and constant link delay [4] as well as lower power consumption than other topologies for application-specific mapping of tasks [5].

Routing algorithms define the path taken by a packet between source and destination switches [6]. They must prevent *deadlock*, *livelock*, and *starvation* [7-8] situations. *Deadlock* may be defined as a cyclic dependency among nodes requiring access to a set of resources, so that no forward progress can be made. *Livelock* refers to packets circulating the network without ever making any progress towards their destination. *Starvation* happens when a packet in a buffer requests an output channel, being blocked because the output channel is always allocated to another packet. Routing algorithms can be classified according to the three different criteria [6]: (i) where the routing decisions are taken; (ii) how a path is defined, and (iii) the path length.

According to where routing decisions are taken, it is possible to classify the routing in *source* and *distributed* routing. In *source* routing, the whole path is decided at the source switch, while in *distributed* routing each switch receives a packet and defines the direction to send it. In source routing, the header of the packet has to carry all the routing information, increasing the packet size [8]. In distributed routing, the path can be chosen as a function of

the network instantaneous traffic conditions. Distributed routing can also take into account faulty paths, resulting in fault tolerant algorithms.

According to how a path is defined, routing algorithms can be classified as *oblivious* and *adaptive* [8]. In *oblivious* routing, the path is completely determined by the source and the destination address. *Deterministic* routing is a subset of oblivious routing, where the same path is always chosen between a source-destination pair. In *adaptive* routing [9], given a source and a destination address, the path taken by a particular packet is dynamically adjusted depending on, for instance, network congestion. With this dynamic load balancing, adaptive routing can potentially achieve better throughput and latency compared to oblivious routing. However, adaptive routing methods face a difficult challenge in balancing router complexity with the capability to adapt. To achieve the best performance through adaptivity, a router ideally needs global knowledge of the current network status. However, due to router speed and complexity, dynamically obtaining a global and instantaneous view of the network is often impractical. Hence, adaptive routing in practice relies primarily on local knowledge, which limits its effectiveness [9]. Though the adaptive routing algorithms can improve the network performance, it adds an overhead in terms of power consumption [10].

According to the path length criterion, routing can be *minimal* or *nonminimal* [7-8]. *Minimal* routing algorithms guarantee shortest paths between source and destination addresses. In *nonminimal* routing, the packet can follow any available path between source and destination. Nonminimal routing offers great flexibility in terms of possible paths, but can lead to livelock situations and increase the latency to deliver the packet.

This paper presents a Power efficient, Oblivious, Path-diverse, Minimal routing technique (POPM) for mesh-based Networks-on-Chip. In this routing technique, routing decision is distributed among all nodes constituting the minimal-path rectangle between the source and destination nodes. The traffic between each source-destination pair is divided in each intermediate node according to the number of minimal paths from the next hop of this intermediate node to the destination. POPM is compared to North-Last routing as one of the turn model routing algorithms [11-13], XY routing as a dimension-order routing (DOR) algorithm [14-15], and finally to Path-based, Randomized, Oblivious, Minimal (PROM) routing algorithm [16]. A cycle-accurate simulation of a 2D-mesh network-on-chip is performed. A set of standard synthetic traffic patterns, namely *shuffle*, *transpose*, *bit-rotation*, and *bit-reversal* as well as a real application traffic pattern for the MPEG4 decoder is applied to the network. For each traffic pattern, throughput, latency, and power consumption are studied. Experimental results show that POPM offers competitive performance and power consumption under various traffic patterns.

The rest of the paper is organized as follows. Section 2 provides a brief overview of related routing algorithms. In Section 3, a review of the related work is presented. The proposed POPM routing and its implementation cost are described in Section 4. Section 5 reports experimental results for both synthetic and real traffic scenarios. Finally, Section 6 concludes the paper and outlines some directions for future work.

2. Overview of Routing Algorithms

In this section, we give a brief overview of the most related routing algorithms for the sake of qualitative comparisons with our proposed routing technique.

2.1 Dimension-Order Routing (DOR)

Dimension order routing (DOR) [15] is a typical minimal turn algorithm. XY routing is a dimension ordered routing which routes packets first in x- or horizontal direction to the correct column and then in y- or vertical direction to the receiver. Addresses of the routers are their xy-coordinates. XY routing suits well on a network using mesh or torus topology. Figure 1 shows an example of XY routing. Figure 2 shows the allowed turns in the XY routing.

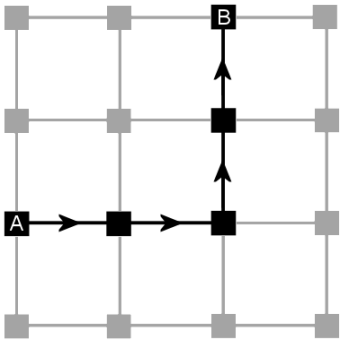


Fig. 1 XY routing from router A to router B

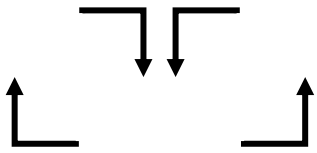


Fig. 2 Allowed turns in XY routing

Though XY routing never runs into deadlock or livelock [17], it does not extend the traffic load regularly over the whole network [18].

2.2 North-Last Routing

A packet passing between switches in a 2D mesh can follow four directions: East, West, North, and South. Eight distinct turns are possible in the path followed by a packet as shown in Figure 3.

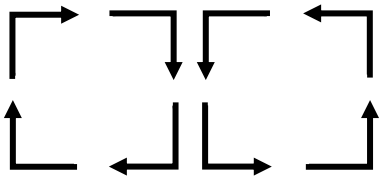


Fig. 3 All possible turns in 2D mesh

Algorithms with no restrictions on turns are named fully adaptive; otherwise they are named partially adaptive. Fully adaptive routing algorithms are subjected to deadlock conditions. Glass and Ni [13] show that, if at least two turns are forbidden, it is possible to implement deadlock free algorithms. This is a sufficient condition for achieving freedom of deadlock. According to the turn model [13], there are four routing algorithms, one deterministic (*XY*) and three partially adaptive (*West-first*, *North-last* and *Negative-first*).

In *North-Last* routing algorithm**,** turns away from north are not possible. Thus the packets which need to be routed to north must be transferred there at last. Figure 4 shows the allowed turns in the north-last routing.

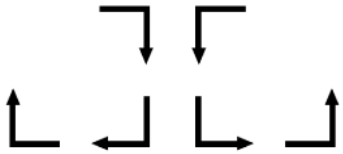


Fig. 4 Allowed turns in north-last routing

2.3 PROM Routing

Given a flow from a source to a destination, PROM [16] routes each packet separately via a path randomly selected from among all minimal paths. The routing decision is made lazily: that is, only the next hop (conforming to the minimal-path constraint) is randomly chosen at any given switch, and the remainder of the path is left to the downstream nodes.

Variable Parameterized PROM (PROMV): PROM algorithm can be parameterized by a single parameter *f*, as shown in Figure 5. At the source node, the router forwards the packet towards the destination on either the horizontal link or the vertical link randomly according to the ratio x+ *f* : y+ *f*, where x and y are the distances to the destination along the corresponding axes. At intermediate nodes, two possibilities exist: if the packet arrived on an X-axis ingress, the router uses the ratio of x+ *f* : y in randomly determining the next hop, while if the packet arrived on an Y-axis ingress, it uses the ratio x : y+ *f* . Intuitively, PROM is less likely to make extra turns as *f* grows up, and increasing *f* pushes traffic from the diagonal of the minimal-path rectangle towards the edges.

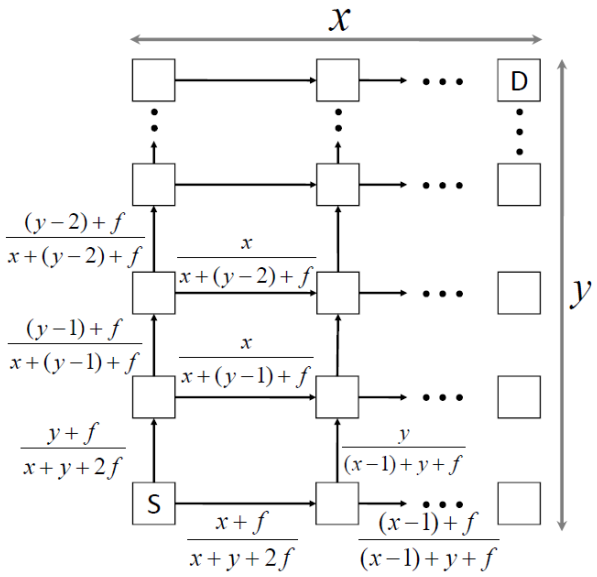


Fig. 5 Parameterized PROM

Variable Parameterized PROM (PROMV) [16] sets the parameter *f* proportional to the minimal-path rectangle size divided by overall network size so traffic can be routed more toward the boundary when the minimal-path rectangle is large. When x and y are the distances from the source to the destination along the X and Y axes and N is the total number of router nodes, *f* is determined by the following equation:

$$f = f_{max} \cdot \frac{xy}{N} \quad (1)$$

Virtual Channel Assignment: PROMV avoids deadlock through appropriate virtual channel assignment, utilizing an observation first made in [19]. The key observation is that minimal-path traffic always obeys one of those two turn models: eastbound packets never turn westward, westbound packets never turn eastward, and packets between nodes on the same row or column never turn at all. PROMV requires only two virtual channels for deadlock-free routing. The virtual channel assignment depends on

the relative position of the source node *S* and destination node *D*. Virtual channels assignment are as follows [16]:

1. If *D* lies to the east of *S*, vertical links use the first VC.
2. If *D* lies to the west of *S*, vertical links use the second VC.
3. If *D* lies directly north or south of *S*, both VCs are used.
4. All horizontal links may use all VCs.

When there are more than two virtual channels, they are split into two sets and assigned similarly.

3. Related Work

Dimension–order routing is usually used for meshes and hypercubes. The basic idea of this routing algorithm is that it routes data packets by crossing dimensions in strictly increasing or decreasing order, reducing the offset to zero in one dimension before routing in the next one [10]. Though the DOR algorithm is easy to implement and has low overhead, its throughput can be poor even for local traffic since it offers no routing flexibility.

Valiant and Brebner [20] proposed one of the best known randomized algorithms named Valiant. This algorithm has two phases. In both of the phases it uses dimension-order routing. In the first phase a random node is selected, and a packet is sent there. In the second phase, it routes the packet from that random node to its destination. As Valiant is non-minimal and tries to avoid congestion in the network, it produces higher packet latency and power consumption.

Another routing algorithm which uses randomization is Randomized, Oblivious, Multi-phase, Minimal (ROMM) [21-22] algorithm. Its minimal nature comes from DOR, and randomization from Valiant. ROMM can have two to n phases in an n × n mesh, with each of the two phases (i.e., from source node to intermediate node and from intermediate node to destination node) may use some variation of DOR (i.e., XY-order or YX-order) [16]. All intermediate nodes must be within the minimum rectangle defined by the source and destination nodes. If n phases are used then ROMM algorithm requires n-virtual channels for wormhole routed mesh network, in order to avoid deadlocks. While increasing the number of phases increases load balancing, it comes at the cost of increased hardware complexity, for example, more virtual channels are required. Also, additional control logic is required to manage virtual channels and their assignment to different phases. As POPM distribute traffic equally among all possible minimal paths between each source-destination pair, it becomes significantly more efficient in its hardware implementation.

O1TURN [23] is a path-diverse routing algorithm. In O1TURN a network is partitioned into two virtual networks. One of them is XY-routed the other is YX-routed. When a packet is injected into the network it randomly choose one of the virtual networks. According to [23] O1TURN outperforms ROMM, Valiant and DOR under non-uniform traffic. But DOR is still better under uniform traffic pattern. POPM routing is better than O1TURN in terms of load balancing, since its path diversity is much greater than that in O1TURN routing.

PROMV [16] is a Path-based, Randomized, Oblivious, Minimal routing algorithm. In PROMV, the probability of each egress being chosen (as well as the value of the parameter f) only depends on the location of the current node and on the relative locations of the source and destination nodes. It sets the parameter f proportional to the minimal-path rectangle size divided by overall network size. For each node to adaptively control its traffic distribution, it must adjust the value of the parameter f. Dynamically adjustment of the value of f requires local intelligence embedded in each node and this will be very costly in terms of control logic hardware. If the value of f will be static, it will be calculated using equation 1 as previously described in section 2. In this equation, PROMV did not specify how the value of f_{max} is justified and how this value will affect the routing decision.

Adaptive routing schemes include turn routing algorithms such as North-Last routing [15]. These are general schemes that allow packets to take different paths through the network while ensuring deadlock freedom but do not specify the mechanism by which a particular path is selected [16]. An adaptive routing policy determines what path a packet takes based on network traffic. POPM algorithm is oblivious routing algorithm which distributes all traffic uniformly among all possible minimal paths in order to balance traffic load among all network nodes. POPM hardware cost is quite simple as the dimension-order routing algorithm. POPM routing achieves better performance and efficient power consumption in comparison with DOR, North-Last, and PROMV routing algorithms.

4. The Proposed Algorithm

POPM is a Power efficient, Oblivious, Path-diverse, Minimal routing technique for mesh-based Networks-on-Chip. For each source-destination pair, POPM distribute traffic equally among all possible minimal paths. POPM makes routing decisions locally at each hop rather than establishing a fixed and deterministic path between the source and destination nodes. The complete POPM algorithm is shown in Figure 6.

POPM Routing Algorithm

```
Let n the number of PEs
Let P an n x n matrix which contains number of packets between each
source-destination pair passing through the current node
Let loc the location of the current node
Let src the location of the source node
Let dest the location of the destination node
Let next_hop_N the location of the next hop in the north direction
Let next_hop_S the location of the next hop in the south direction
Let next_hop_E the location of the next hop in the east direction
Let next_hop_W the location of the next hop in the west direction
Let output the output direction of the packet

//initialization
for i=1 to n do
        for j=1 to n do
                P[i][j]=0
        end for
end for
if (dest.x == loc.x or dest.y == loc.y) begin
        output=routing_XY(loc, dest)
end
else begin
        tot_paths = get_num_of_paths(loc, dest)
        remainder = P[src][dest] % tot_paths
        next_hop_ N.x = loc.x
        next_hop_ N.y = loc.y - 1
        next_hop_ S.x = loc.x
        next_hop_ S.y = loc.y + 1
        next_hop_E.x = loc.x + 1
        next_hop_E.y = loc.y
        next_hop_W.x = loc.x - 1
        next_hop_W.y = loc.y
        if (dest.x > loc.x and dest.y < loc.y) begin
                x_paths = get_num_of_paths(next_hop_E, dest)
                y_paths = get_num_of_paths(next_hop_ N, dest)
                if(remainder < x_paths) begin
                        output=DIRECTION_EAST
                end
                else begin
                        output=DIRECTION_NORTH
                end
        end
        else if (dest.x > loc.x and dest.y > loc.y) begin
                x_paths = get_num_of_paths(next_hop_E, dest)
                y_paths = get_num_of_paths(next_hop_ S, dest)
                if(remainder < x_paths) begin
                        output=DIRECTION_EAST
                end
                else begin
                        output=DIRECTION_SOUTH
                end
        end
        else if (dest.x < loc.x and dest.y > loc.y) begin
                x_paths = get_num_of_paths(next_hop_W, dest)
                y_paths = get_num_of_paths(next_hop_ S, dest)
                if(remainder < x_paths) begin
                        output=DIRECTION_WEST
                end
                else begin
                        output=DIRECTION_SOUTH
                end
        end
        else begin
                x_paths = get_num_of_paths(next_hop_W, dest)
                y_paths = get_num_of_paths(next_hop_ N, dest)
                if(remainder < x_paths) begin
                        output=DIRECTION_WEST
                end
                else begin
                        output=DIRECTION_NORTH
                end
        end
        P[src][dest]++
end
```

Fig. 6 POPM routing algorithm

For each packet passing through an intermediate node there are three cases:

1) Destination node is at the same row.
2) Destination node is at the same column.
3) Destination node is at different row and different column.

For cases 1 and 2, we simply use XY routing. For case 3, the traffic between each source-destination pair is divided in each intermediate node according to the number of minimal paths from the next hop of this intermediate node to the destination. Figure 7 illustrates an example for traffic distribution between a source and destination nodes using POPM algorithm.

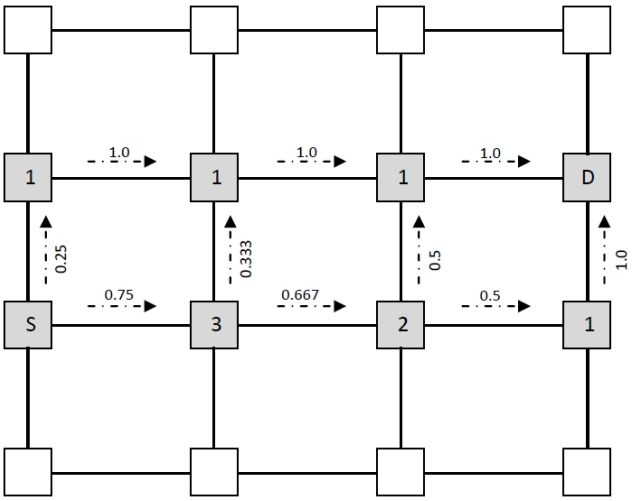


Fig. 7 Traffic distribution between a source and destination nodes using POPM routing algorithm

All packets generated by the source node S and destined to the destination node D will be routed to either the east or the north direction to preserve the minimal path between S and D. The number inside each node in the minimum rectangle between S and D represents the number of minimal paths from this node to D. At the source node S, there are four minimal paths to the destination node D; three of them are through the next hop in the east direction

and only one through the next hop in the north direction. So, 75% of the packets destined to D will be routed to the east direction while 25% will be routed to the north direction. The packets will be distributed at each intermediate node until finally reach their destination node. Note that POPM routing is oblivious and next-hop routing decisions can be computed locally at each node based on local information and the relative position of the current node to the destination node; nevertheless, the algorithm is maximally diverse in the sense that each possible minimal path has an equal probability of being chosen.

POPM calculate *tot_paths*, which is the total number of paths between the current node and the destination node. Then the algorithm will divide the total number of packets between the source node and destination node which passed through the current node by the *tot_paths* and calculate the *remainder*. The algorithm calculates the location of the next hop in the four directions. According to the location of the current node relative to the location of the destination node, POPM determines the two possible directions to which it can route the current packet ((East-North), (East-South), (West-North), and (West-South)). The number of paths between the next hop and the destination node in the X and Y dimensions is calculated and stored in *x_paths* and *y_paths* respectively. So, according to the value of the *remainder*, POPM selects one of the two possible hops to route the current packet to it. POPM avoids deadlock using the same virtual channel allocation scheme used in [16] and described in section 2.

POPM Implementation Cost: POPM implementation does not require any special hardware overhead over any simple oblivious virtual channel router [15]. In POPM, as with DOR, the choice of the egress port depends only on the location of the current node relative to the destination node. So, there is no additional overhead required in the packet header. Virtual channel allocation also requires only local information already available in the classical router: namely, the ingress port and ingress VC must be provided to the VC allocator and constrain the choice of available VCs when routing to vertical links, which, at worst, requires simple multiplexer logic [16]. This approach ensures deadlock freedom, and eliminates the need to keep any extra routing information in packets. Thus, POPM needs equivalent hardware to that of the simplest DOR routing algorithm.

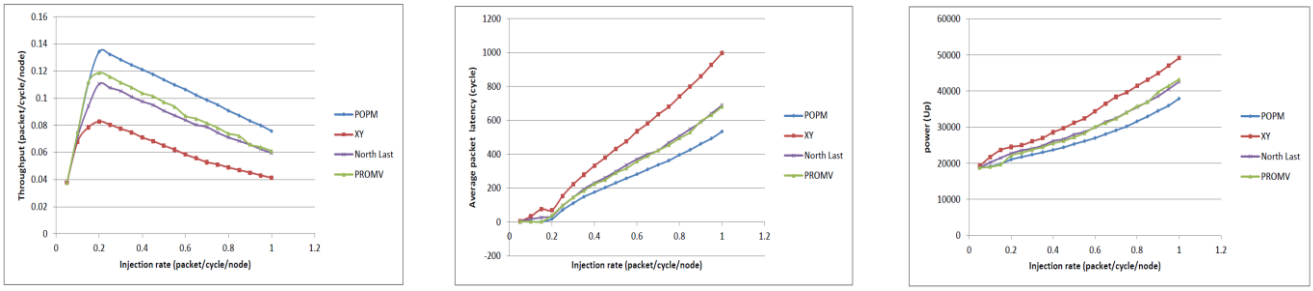
5. Experimental Results

In this section, the performance of POPM routing algorithm is to be evaluated and compared to DOR, North-Last, and PROMV routing algorithms. All algorithms were applied to a NoC with a 2D-mesh topology. A system consisting of 16 Intellectual Property (IP) blocks mapped onto Mesh-based NoC architecture was considered. We characterize the performance of the NoC under consideration in terms of throughput, latency and power consumption. A cycle-accurate simulator [24] is used to examine the actual performance and power consumption on common virtual channel router architecture. A set of standard synthetic traffic patterns, namely *shuffle*, *transpose*, *bit-rotation*, and *bit-reversal* as well as a real application traffic pattern for the MPEG4 decoder is applied to the network. These traffic patterns do not reflect all traffic on an arbitrary network; nevertheless, they were designed to simulate traffic produced by real-world applications [15], and so are often used to evaluate routing algorithm performance [16]. In our simulation, routers were configured for 8 virtual channels per port, allocated either in one set (for DOR and North-Last) or in two equal sets (for POPM and PROMV), and then dynamically within each set. Table 1 summarizes all network configurations.

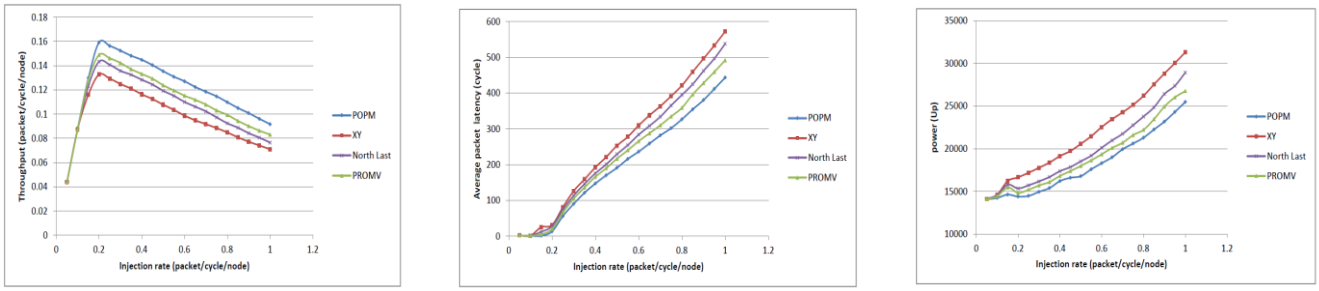
Table 1: Network Parameters Configuration

Parameter	Configuration
Topology	2D Mesh
DimX	4
DimY	4
NUM_INPUTS	5
VC_NUM	8
VC_BUFFER_SIZE	4
PACKET_INJECTION_RATE	0.05 to 1 with step 0.05
TRAFFIC_DISTRIBUTION	*shuffle*, *transpose*, *bit-rotation*, *bit-reversal*, and *MPEG4 traffic*
ROUTING_ALGORITHM	*DOR(XY), North-Last, PROMV, and POPM*
PACKET_SIZE	4 *flits*
WARM_UP_TIME	1000
SIMULATION_TIME	10000

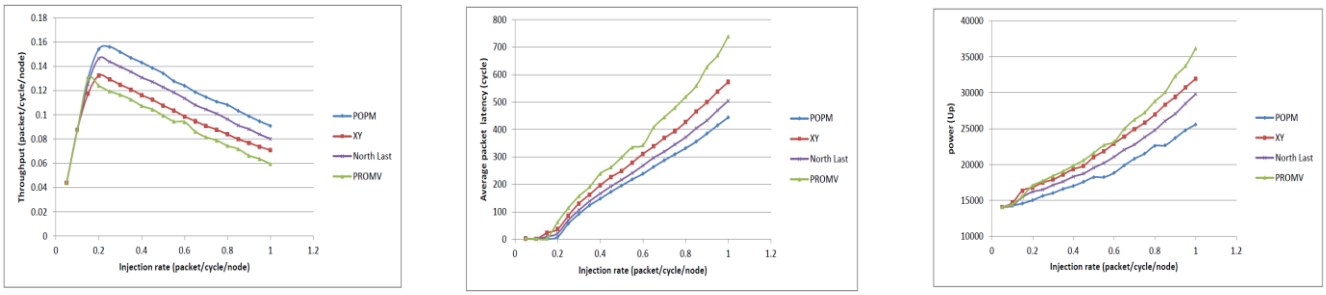
Simulation results for throughput, latency, and power for the different synthetic traffic patterns are shown in Figure 8. Note that power consumption is given in unit power (U_p) [24]. As shown, POPM shows better throughput, latency, and power consumption than DOR, North-Last, and PROMV under transpose, shuffle, and bit-rotation traffic patterns. In bit-reversal traffic pattern, POPM outperforms DOR and North-Last, but it gives slightly better performance than PROMV. Also, in bit-reversal PROMV gives comparable power consumption with respect to POPM. The perfect symmetry of bit-reversal traffic causes congestion to be directly proportional to the degree to which the network load is balanced among nodes.



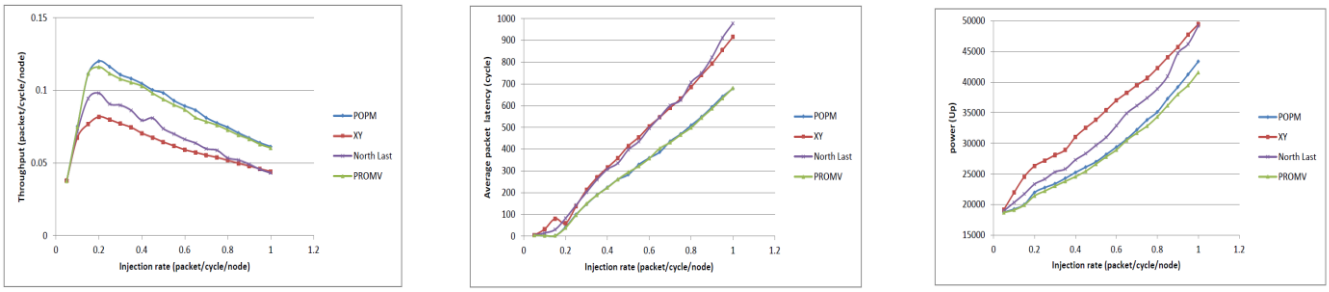
(a) Transpose Traffic



(b) Shuffle Traffic



(c) Bit-Rotation Traffic



(d) Bit-Reversal Traffic

Fig.8 Throughput, Latency, and Power for the different synthetic traffic patterns

So, and as POPM is maximally minimal path diverse algorithm, it consumes more power since the number of packets which are injected to the network but does not reach their final destination node is increased. Simulation results for throughput, latency, and power for the MPEG4 real traffic pattern is shown in Figure 9. The x-axis shows the packet injection rate (PIR) scale factor used for the simulation. This PIR scale factor indicates the scaling factor used for injecting traffic into the network. Thus, a PIR scale factor of one indicates that the total volume of injected traffic during the simulation is the sum of all the values in the table of communication requirements. A PIR scale factor of two indicates that each element of the table has been doubled, and therefore the volume of the injected traffic during the simulation has been doubled. As shown in Figure 9, POPM gives better performance and power saving especially for higher traffic rates than all other algorithms. As shown POPM offers competitive performance and efficient power consumption under a variety of traffic patterns because it can distribute traffic load among many network links.

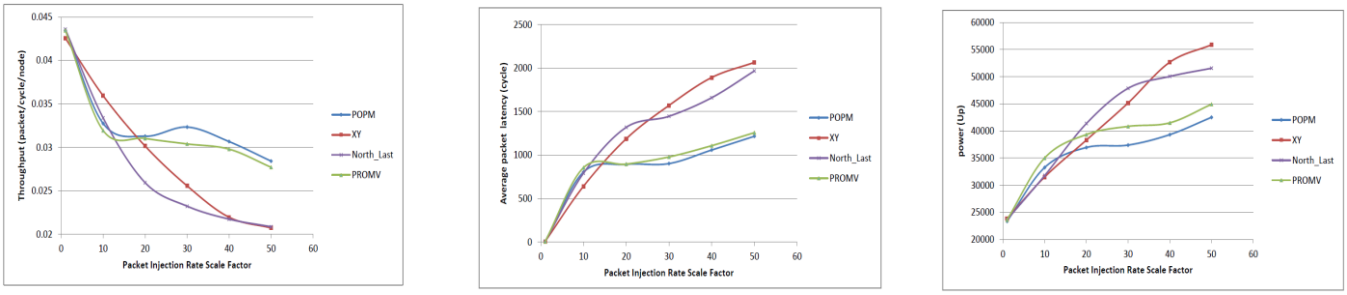


Fig.9 Throughput, Latency, and Power for the MPEG4 traffic pattern

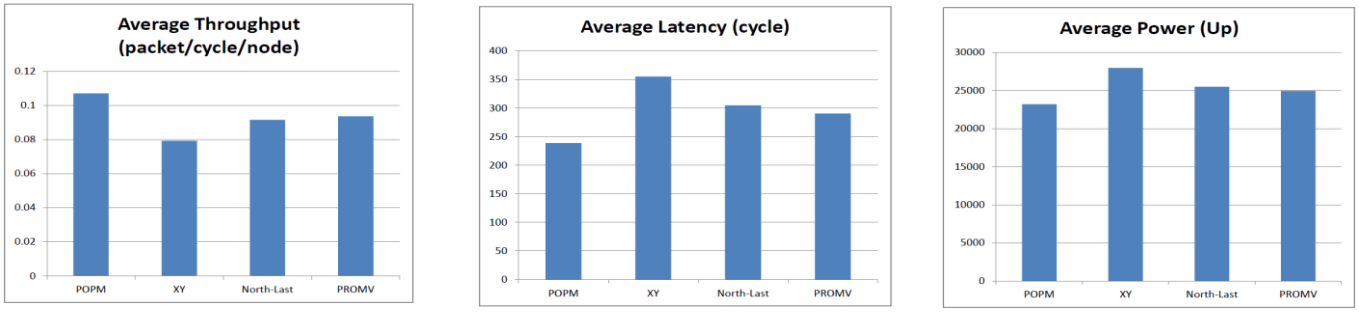


Fig. 10 Average Throughput, Latency, and Power for POPM, XY, North-Last, and PROM under all traffic patterns

Also, average throughput, average latency, and average power consumption for all algorithms under all traffic patterns used in simulation are calculated as shown in Figure 10. Based on these results, POPM offers 15% better average throughput, 20% less average latency, and 10% less average power consumption than all other algorithms.

6. Conclusions

In this paper, a Power efficient, Oblivious, Path-diverse, Minimal routing technique (POPM) for mesh-based Networks-on-Chip is developed. POPM achieves maximum path diversity in all minimal routing algorithms by uniformly distributing network traffic among all nodes constituting the minimal-path rectangle between the source and destination nodes. POPM hardware cost is quite simple as the dimension-order routing algorithm.

A cycle-accurate simulation under a set of standard synthetic traffic patterns (*shuffle*, *transpose*, *bit-rotation*, and *bit-reversal*) as well as a real application traffic pattern for the MPEG4 decoder is conducted. Results show that, POPM routing achieves better performance and efficient power consumption in comparison with DOR, North-Last, and PROMV routing algorithms under various traffic patterns. Also, we would expect POPM to offer higher performance on most traffic patterns because it shows 15% better average throughput, 20% less average latency, and 10% less average power consumption than all other algorithms. POPM can produce better performance if the blocked packet (due to traffic congestion) can change its direction after a specific amount of blocking time depending on a specific threshold value. This topic is left for future work.

References

[1] L. Benini et al., "Networks on Chips: A New SoC Paradigm", Computer, vol. 35, no. 1, Jan. 2002, pp. 70-78.

[2] P. Pande, C. Grecu, A. Ivanov, R. Saleh, G. De Micheli, “Design, Synthesis and Test of Networks on Chip”, IEEE Design and Test of Computers, Vol. 22, No. 5, 2005. pp: 404-413.

[3] W.J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks", Proc. 38th Design Automation Conf., ACM Press, 2001, pp. 681-689.

[4] R. Tornero, J. M. Orduña, A. Mejia, J. Flich and J. Duato, " A Communication-Driven Routing Technique for Application-Specific NoCs ", International Journal of Parallel Programming, Vol. 39, No. 3, June 2011, pp. 357-374.

[5] El Sayed M. Saad, Medhat H. Awadalla, Sameh A. Salem and Ahmed M. Mostafa, " Network-on-Chip: Power Optimization Architecture Mapping based on Global Interconnection Links", International Journal of Computer and Electrical Engineering (IJCEE), Vol. 3, No 2, April 2011, pp. 289- 296.

[6] A.V.de Mello, L.C.Ost, F.G.Moraes, and N.L.V.Calazans, "Evaluation of Routing Algorithms on Mesh Based NoCs", PUCRS, Av.Ipiranga, 2004.

[7] J. Liang, S. Swaminathan, and R. Tessier, "aSOC: A Scalable, Single-Chip communications Architecture", In IEEE International Conference on Parallel Architectures and Compilation Techniques, Oct. 2000, pp. 37-46.

[8] L. M. Ni and P. K. McKinley, "A Survey of Wormhole Routing Techniques in Direct Networks", IEEE Computer Magazine, v.26(2), February, 1993, pp. 62-76.
[9] M.A. Kinsy, M.H. Cho, T. Wen, G.E. Suh, M.V. Dijk, and S. Devadas, "Application-aware deadlock-free oblivious routing", in Proc. ISCA, 2009, pp.208-219.
[10] H. Zhu, P. P. Pande, and C. Grecu, "Performance evaluation of adaptive routing algorithms for achieving fault tolerance in NoC fabrics", in IEEE International Conf. on Application-specific Systems, Architectures and Processors, ASAP 2007, Montreal, Que., Jul. 2007, pp. 42–47.
[11] C.J. Glass and L.M. Ni, "The Turn Model for Adaptive Routing", Proceedings of the 19th Annual International Symposium on Computer Architecture, May 1992. pp: 278-287.
[12] C.J. Glass and L.M. Ni, "Adaptive Routing in Mesh-connected Networks", Proceedings of the 12th International Conference on Distributed Computing Systems, June 1992. pp: 12-19.
[13] G. J. Glass and L. M. Ni, "The turn model for adaptive routing", Journal of ACM, 40, (5), Sept. 1994, 874-902.
[14] J. Duato, S. Yalamanchili, L. Ni, "Interconnection Networks an Engineering Approach", IEEE Computer Society, 2003.
[15] W.J. Dally, B. Towles, "Principles and Practices of Interconnection Networks", Morgan Kaufmann, 2004.
[16] M. H. Cho et al., "Path-based, randomized, oblivious, minimal routing", in International Workshop on Network on Chip Architectures, 2009, pp. 23–28.
[17] M. Dehyadgari, M. Nickray, A. Afzali-kusha, Z. Navabi, "Evaluation of Pseudo Adaptive XY Routing Using an Object Oriented Model for NOC", The 17th International Conference on Microelectronics, December 2005.
[18] Ville Rantala, Teijo Lehtonen, and Juha Plosila, "Network on Chip Routing Algorithms", TUCS Technical Report, No. 779, August 2006.
[19] K. S. Shim, M. H. Cho, M. Kinsy, T. Wen, M. Lis, G. E. Suh, and S. Devadas, "Static Virtual Channel Allocation in Oblivious Routing", In Proceedings of the 3rd ACM/IEEE International Symposium on Networks-on-Chip, May 2009.
[20] L. G. Valiant and G. J. Brebner, "Universal schemes for parallel communication", In STOC ’81: Proceedings of the thirteenth annual ACM symposium on Theory of computing, 1981, pp. 263–277.
[21] T. Nesson and S. L. Johnsson, "ROMM Routing: A Class of Efficient Minimal Routing Algorithms", In in Proc. Parallel Computer Routing and Communication Workshop, 1994, pp. 185–199.
[22] T. Nesson and S. L. Johnsson, "ROMM routing on mesh and torus networks", In Proc. 7th Annual ACM Symposium on Parallel Algorithms and Architectures SPAA’95, 1995, pp. 275–287.
[23] D. Seo, A. Ali, W.-T. Lim, and N. Rafique, "Near-optimal worst- case throughput routing for two-dimensional mesh networks", Proceedings of 32nd International Symposium on Computer Architecture, ISCA '05, 2005, pp. 432 - 443.
[24] El Sayed M. Saad, Sameh A. Salem, Medhat H. Awadalla and Ahmed M. Mostafa, "PPNOCS: Performance and Power Network on Chip Simulator based on SystemC", International Journal of Computer Science Issues (IJCSI), Vol. 8, Issue 6, No 3, November 2011, pp. 169-179.